UCRL-MI-142491

Parallel FTP Performance in a High-Bandwidth, High-Latency WAN

Jason S. King

U.S. Department of Energy

Lawrence Livermore National Laboratory

November 10, 2000

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

Background

Within the Accelerated Strategic Computing Initiative (ASCI) program, the Distance and Distributed Computing program, or DisCom^2, is charged with ensuring that Laboratory scientists have the best possible access to computing resources no matter where those resources may be located. Network file transfer has been identified as an important part of this effort. The three weapons labs, Lawrence Livermore, Sandia, and Los Alamos, have been working for some time on plans for a secure, high-speed, low-latency Wide Area Network (WAN) spanning the sites in Livermore, Albuquerque, and Los Alamos. The proposed file transfer tool for this new network is the parallel file transfer protocol (FTP) client as distributed with the High Performance Storage System (HPSS).

This tool was chosen because it is a mature code that has many of the desired capabilities identified by the DisCom^2 program. Among these capabilities are parallel file transfer, compatibility with HPSS, and backward compatibility with "standard" FTP as specified by RFC 959.

Testing at LLNL in early September 2000 showed that, although the parallel FTP code meets the desired capability requirements, its performance in the Local Area Network (LAN) is not quite optimal. Examination of the architecture revealed that a large portion of the performance penalty is directly related to the code's support of the mover and pdata protocols needed to communicate with HPSS. These protocols basically introduce a lock step for each block of data sent across the network. While this did not appear to significantly reduce performance in the LAN, the higher latency found in the WAN would likely result in greatly decreased performance. At least one of these two protocols will always remain necessary for communication with HPSS. They are not, however, required when communicating between two non-HPSS systems, for example, between ASCI White and an SGI visualization platform.

LLNL has demonstrated that, in the absence of HPSS, parallel file transfer can be accomplished with much less overhead and higher performance, even in the LAN. A modification was made to the parallel FTP client and server (non-HPSS server based on the public domain wuftpd code and parallel modifications from M. Barnaby at Sandia) that essentially removed the mover and pdata protocols in favor of a much simpler protocol with 16 bytes of overhead per parallel stripe, per file transfer, with no lock-step mechanism.

Considering the importance of file transfer performance to the upcoming DisCom^2 WAN, it was decided that SC2000, held in early November, would provide a great opportunity for testing the two versions of parallel FTP. For purposes of the discussion in the rest of this paper, the standard HPSS version of parallel FTP, which includes the mover and pdata protocols, is referred to as the "PFTP-hpss," while the modified version is referred to as "PFTP-simple."

SC2000 Network Topology

Source and sink hosts were identified that adequately represent the actual platforms in use on ASCI networks today. Those hosts were an SGI Onyx2 located on the SC2000 show floor in Dallas and an IBM Nighthawk-1 SP node located in Livermore. Each host was connected to the network with four Gigabit Ethernet adapters. Each adapter was placed in a separate Virtual LAN (VLAN), and all traffic was carried across a 2.5 Gb/s OC48c between Livermore and Dallas. The network topology is shown in Figure 1.

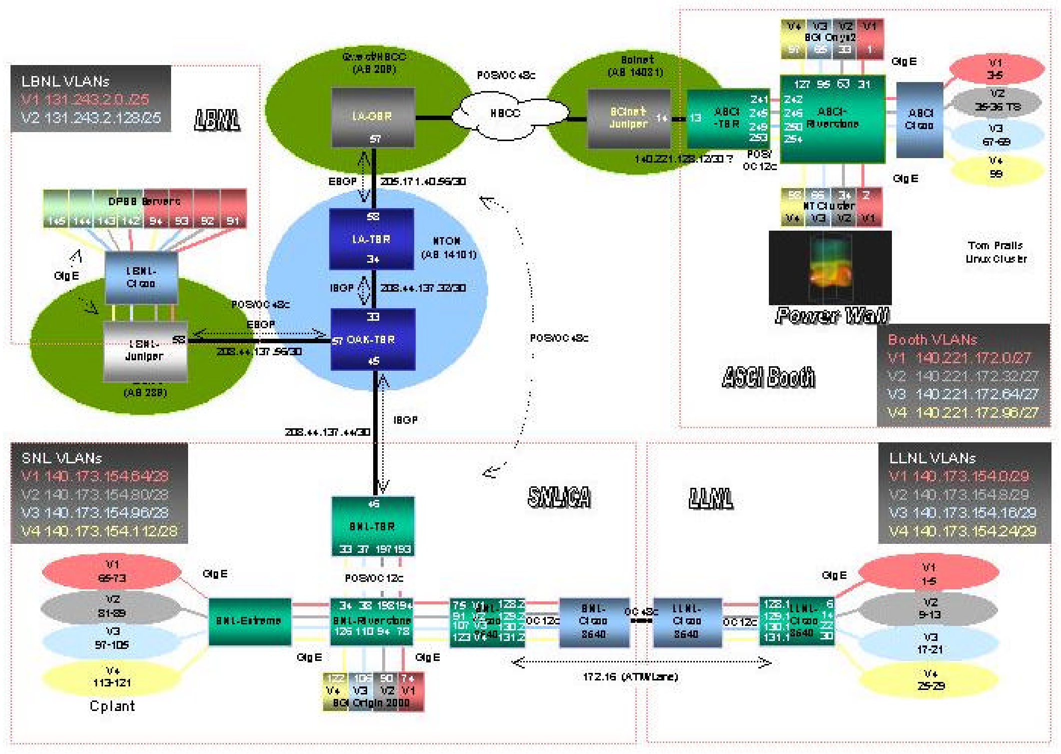

Figure 1. Network topology for testing two versions of parallel FTP.

Tests

The testing methodology was first to use the netperf benchmark tool (http://www.netperf.org) to establish a performance baseline for the network between the source and sink. Because it does not perform disk I/O and strictly measures memory-to-memory copies over TCP and UDP, netperf is a good tool for determining maximum network performance between two hosts. Then, equipped with the knowledge of what was theoretically possible on the network, we looked at how the two different parallel transfer methods performed in comparison.

Iterations of netperf were used to determine the optimal TCP window size to tune the network for maximum performance. With the hosts configured for maximum performance, each transfer method, PFTP-hpss and PFTP-simple, was run through iterations of different parameters, including block sizes and stripe widths. Packet traces of a typical transfer were generated for each method for additional analysis.

Results

The architecture of the network connecting LLNL to the SC2000 show floor was such that, to take full advantage of the bandwidth available, it was necessary to have a minimum of four TCP streams, each destined for a different subnet on the remote end. This allowed traffic to be spread across each of four

OC-12 ATM links. Since the round-trip delay was 50 ms and OC-12 bandwidth is 622 Mb/s, the bandwidth delay product of this network is 3.88 MB. Under optimal conditions, the bandwidth delay product should yield the optimal TCP window size, but in this case it did not. Increasing the TCP window size beyond 2 MB had an adverse effect on performance. Unfortunately, the results above 2 MB are not available, but Figure 2 shows that performance plateaus at a 1.5 MB window size. It is believed the poor performance beyond the 2 MB window is directly related to TCP's slow start and congestion control algorithms. Because this network would not run at full bandwidth without losing packets, anytime the TCP window started to approach the network's maximum speed, a packet was lost, TCP's congestion control algorithm activated, and performance declined. TCP appeared to take a very long time to reopen the window once packet loss occurred.

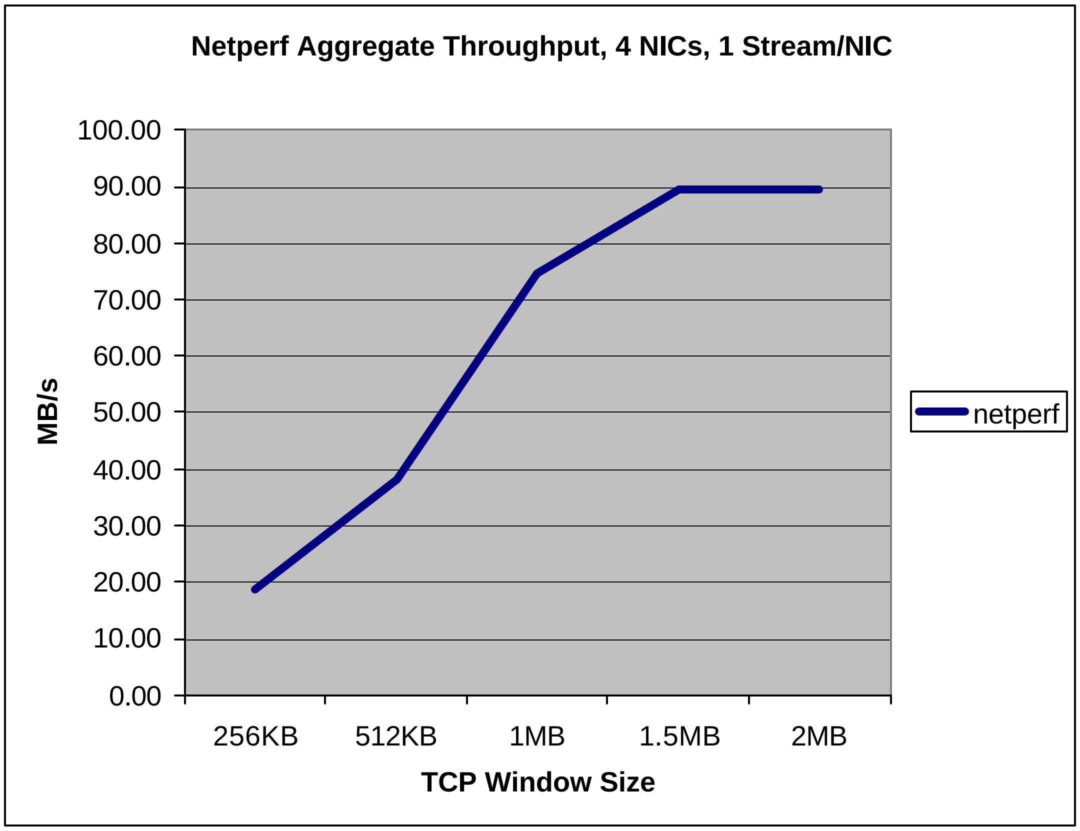

Figure 2. The netperf aggregate throughput performance.

Based upon the results shown in Figure 2, both hosts were configured with socket buffers of 2 MB for the remainder of the tests. Figure 3 illustrates the performance of netperf and the two parallel FTP methods when using 2 MB socket buffers. The performance of the PFTP-hpss method, in general, was half that of the PFTP-simple method.

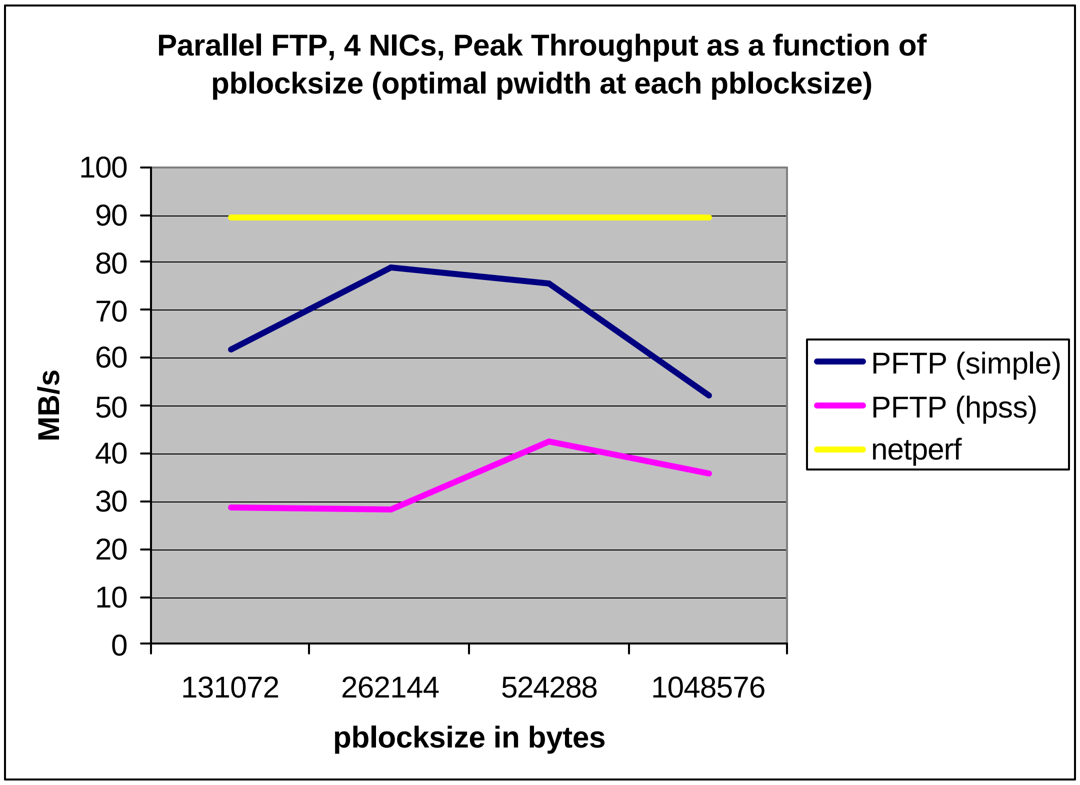

Figure 3. The throughput performance of netperf and the two parallel FTP methods.

Discussion

Figures 4 and 5 show why the performance of PFTP-hpss is sub-optimal. Figures 4 and 5 represent a one second snapshot of throughput from one of four total streams. Figure 4 is a good illustration of the impact that the mover protocol has in the WAN. Because PFTP-hpss operates in a lock-step fashion—sending a mover protocol message, then awaiting the acknowledgement before sending data—large gaps are created where throughput drops to zero. These gaps should be roughly the same size as the round-trip delay of 50 ms. Examination of the packet data bears this out.

Figure 5 shows the same data for PFTP-simple. Note that it also fluctuates greatly over time, but in a more rapid fashion, and average throughput is roughly twice that of PFTP-hpss. The relatively small TCP window used likely explains the rapid fluctuations. Because the chosen window size was small enough to avoid packet loss, it does not allow the network to be fully utilized. Figure 5 seems to support this theory in that slightly less than 50% of the total time shown on the graph is spent idling. Because 2 MB is roughly 52% of the optimal window size of 3.88 MB, we would expect to see approximately 48% of the time spent idle.

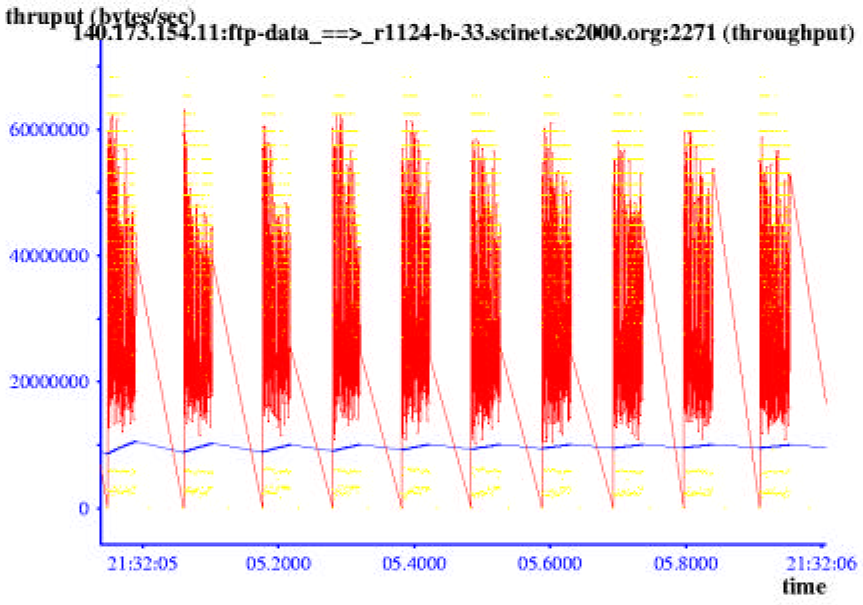

Figure 4. PFTP-hpss, pwidth 4, pblocksize 256 KB.

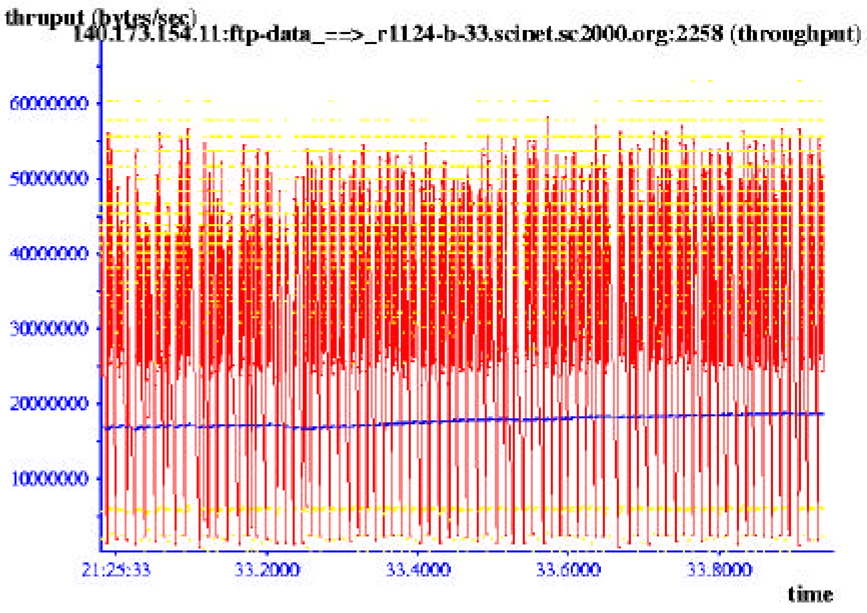

Figure 5. PFTP-simple, pwidth 4, pblocksize 256 KB.

Conclusion

At the least, this work shows that packet loss in the WAN can severely limit throughput. It also shows that there is great room for improvement in our chosen method of file transfer in the WAN and, at least for the moment, that the largest performance gains in the WAN will likely come from work on the protocols in use, rather than from work on disk or system I/O issues. We need protocols capable of high performance in the WAN before we can expect to fully utilize increasingly high bit rate networks.

Acknowledgements

Special thanks to Bryan Lawver (LLNL) and to Helen Chen and Jim Brandt (SNL/CA) for all their work in setting up this network and babysitting it through a hectic week at SC2000. Also, thanks to the many people at SNL/NM who supplied the SGI and helped with administrative issues.